Models Performance

Business Problem Overview and Solution Approach

Business Problem

The cancellation of hotels rooms bookings and reservations causes a lot of disruptions in such a competitive space, which translates to loss of resources and efforts, and more importantly loss of potential revenue that could have been secured otherwise.

So we have a dataset of hotels reservations made in 2017 & 2018 , with various recorded features to help us identify patterns and trends in the data, and to create a Machine Learning model that can also help in predicting whether a reservation is likely to be cancelled or not.

Solution Approach

- Understand the request very carefully.
- Apply a generic feature understanding, by analyzing each feature and its relationship with other relevant features.
- Go through the recommended univariate analysis tasks and questions.
- Go through the recommended bivariate analysis tasks and questions.
- Give a very detailed showcase of the insights and findings resulting from the EDA.
- Train and compare a couple of machine learning models to see which would perform better for this use-case.

Repository Breakdown

- GitHub repository:

```
https://github.com/RainOfAshes/hotels-booking-study
```

- Please refer to the README.md in the repo.
- notebooks/eda_0_generic_features_understanding.ipynb has the generic EDA process.
- eda_1_univariate_bivariate_analyses.ipynb has the answers for the questions highlighted in the slides deck
- documents/ have the relevant documents
- reports/ EDA_report.pdf the report for the EDA
- reports/ models_report.pdf the report for the model training results
- plots_and_figs/ should include the plots and charts resulting from the EDA

Models Training Steps

Feature Engineering & Selection

Based on the results of the EDA and the insights generated, I have engineered and chose the following features for the training:

- `has_special_requests` Binary value of either there is a special request or not.
- Encoded the market segments into `{'Online': 0, 'Offline': 1, 'Corporate': 2, 'Other': 3}` Since we will be using decision trees or ensemble models, then ordinal encoding with some meaning might be beneficial.
- `has_children` Binary value of either there is a child or not.
- `year_quarter` 1, 2, 3, 4 for each quarter in the year.
- `arrival_day_name` name of the day of the arrival
- `total_nights` total nights booked
- `total_guests` total guests, both adults and children

Data splitting

Since I am planning to further fine-tune and prune the trees, I had to split the data into three splits:

- Training 70%
- Validation 20%
- Testing 10%

I have also made sure to stratify the shuffling so that each evaluation set has a consistent distribution of the target label.

Initial Models Training

I have chosen the following models for the initial training stage:

- Decision Tree: Default Parameters, Scikit-Learn implementation
- Random Forest: Default Parameters, Scikit-Learn implementation
- Gradient Boosting: Default Parameters, Scikit-Learn implementation
- Always predict True

- Always predict False

Evaluation Metrics

- Accuracy: Not the most representative due to the imbalance in the target label
- Recall
- Precision
- F1-Score (Harmonic Mean between recall & precision)

Fine-Tuning & Trees Pruning

- I have plotted the trees
- Viewed the weights for each feature
- Reviewed the splitting process and the depths.
- Applied Grid Search over a space of parameters
- Chose the best scorers on the eval set
- Choose the best classification threshold to maximize F1-Score

Models Validation Results

Note: Label y = 1 is Cancelled reservation. and Label y = 0 is Not-Cancelled Reservation

Training Split (Default Threshold 0.5)

precision	recall	f1_score	accuracy	model_name
0.979035	0.966314	0.972633	0.982182	random_forest_default
0.985665	0.959577	0.972446	0.982182	decision_tree_default
0.935714	0.882579	0.908370	0.941659	random_forest_pruned
0.836099	0.800289	0.817802	0.883160	decision_tree_pruned
0.778307	0.707892	0.741431	0.838221	gdbt_default

precision	recall	f1_score	accuracy	model_name
0.621685	0.767084	0.686773	0.770735	decision_tree_pruned_ccp
0.327657	1.000000	0.493587	0.327657	always_ones
0.000000	0.000000	0.000000	0.672343	always_zeros

Validation Split (Default Threshold 0.5)

precision	recall	f1_score	accuracy	model_name
0.796491	0.764310	0.780069	0.858720	random_forest_pruned
0.750000	0.737374	0.743633	0.833333	gdbt_default
0.749141	0.734007	0.741497	0.832230	decision_tree_pruned
0.747368	0.717172	0.731959	0.827815	random_forest_default
0.665672	0.750842	0.705696	0.794702	decision_tree_default
0.618421	0.791246	0.694239	0.771523	decision_tree_pruned_ccp
0.327815	1.000000	0.493766	0.327815	always_ones
0.000000	0.000000	0.000000	0.672185	always_zeros

Validation Split (Threshold to maximize important metrics)

precision	recall	f1_score	accuracy	model_name
0.743976	0.831650	0.785374	0.850993	random_forest_pruned
0.699140	0.821549	0.755418	0.825607	gdbt_default
0.706231	0.801347	0.750789	0.825607	decision_tree_pruned
0.618421	0.791246	0.694239	0.771523	decision_tree_pruned_ccp

Testing Split (Default Threshold 0.5)

precision	recall	f1_score	accuracy	model_name
0.752182	0.725589	0.738646	0.831771	random_forest_pruned
0.699839	0.734007	0.716516	0.809708	decision_tree_pruned

0.728242	0.690236	0.708729	0.814120	gdbt_default
0.585079	0.752525	0.658321	0.744071	decision_tree_pruned_ccp

Testing Split (Threshold to maximize important metrics)

precision	recall	f1_score	accuracy	model_name
0.725705	0.779461	0.751623	0.831219	random_forest_pruned
0.693498	0.754209	0.722581	0.810259	decision_tree_pruned
0.709150	0.730640	0.719735	0.813569	gdbt_default
0.585079	0.752525	0.658321	0.744071	decision_tree_pruned_ccp

Attributes Weights

- For best performing models (pruned random forest and pruned decision tree), the main contributing feature is `lead_time` at 0.5 .
- After that in different order, is the following features: `'has_special_requests'` `'normalized_market_segment_value'` `'arrival_month'` with values between 0.1 and 0.08
- The least weighted attributes are : `'is_repeated_guest'` `'has_car'` `'has_children'` because these binary features are positive only in a very small potion of the data 7% for children and 1% for guests.

Metrics Importance

- Due to the imbalanced distribution of the predicted label, accuracy will be heavily misleading.
- In this case, Recall measures the percentage of actual cancellations (positive cases) that the model correctly identifies which is important for mitigating potential revenue losses or optimizing resources.
- Precision measures the proportion of predicted cancellations that are actually canceled, precision means the model avoids false positives (cases where it predicts a cancellation, but the reservation is actually not canceled).

- F1 score indicates a balance between catching as many cancellations as possible and minimizing false positives.
- In this implementation, we should focus on maximizing the recall, to catch as potential cancellations as possible, because the cost of missing a cancellation is higher than falsely predicting it.

Conclusions & Insights

- The overall best performing model was the pruned Random Forest. Due to the nature of the data and the imbalance between its features, randomly creating the ensemble trees will more likely generalize better than using one decision tree.
- Pruning has helped in reducing overfitting which can be easily noticed from the training scores before the pruning.
- The recall is the most important metric to maximize in our case while keeping the f1-score in mind too.